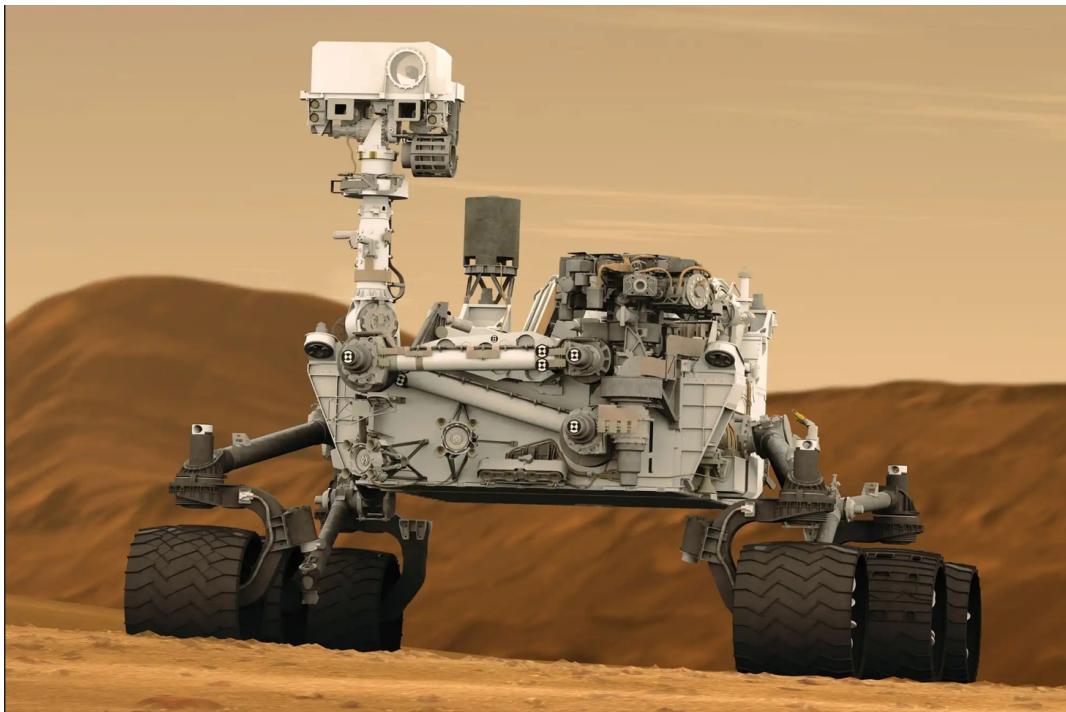


Universidad de Alicante

INGENIERÍA ROBÓTICA

ROBOTS MÓVILES CON RUEDAS ACKERMANN

Trabajo teórico



Autores:

Carmen Losantos

Luis Pérez-Brotóns

Marina Villanueva

15 de enero de 2023

Índice

| | |
|---|-----------|
| 1. Introducción | 2 |
| 2. Objetivos | 2 |
| 3. Configuraciones Cinemáticas de los Robots Móviles | 2 |
| 3.1. Tipos de Robots Móviles con Ruedas | 3 |
| 3.1.1. Ruedas Ackermann | 3 |
| 3.1.2. Ruedas Triciclo clásico | 4 |
| 3.1.3. Ruedas Direcccionamiento diferencial | 4 |
| 3.1.4. Ruedas Skid Steer | 5 |
| 3.1.5. Ruedas Síncronas | 5 |
| 3.1.6. Ruedas Tracción omnidireccional | 6 |
| 4. ¿Qué es y cómo funciona el principio de Ackermann? | 6 |
| 5. Ejemplos de Robots que emplean las ruedas Ackermann | 9 |
| 5.1. Curiosity | 9 |
| 5.2. Knightscope K5 | 10 |
| 5.3. Nomad | 10 |
| 6. Aplicación práctica | 11 |
| 6.1. Race car | 11 |
| 6.1.1. Creación del mundo | 12 |
| 6.1.2. Creación del robot con ruedas Ackermann | 14 |
| 6.1.3. Desarrollo del script path_pursuit.py | 15 |
| 6.1.4. Ejecución del paquete | 16 |
| 6.1.5. Vídeo | 17 |
| 6.2. Simulación de Curiosity | 17 |
| 7. Conclusión | 19 |
| 8. Bibliografía | 20 |

1. Introducción

Los robots pueden moverse de varias maneras, dependiendo de su diseño y de su entorno. Algunos se mueven mediante ruedas o pistones, mientras que otros utilizan patas o tentáculos para desplazarse. Pueden volar o nadar, mientras que otros pueden reptar o caminar sobre terrenos irregulares.

Los robots móviles con ruedas son una de las configuraciones más comunes en la actualidad. Éstos tienen dos o más ruedas y se desplazan mediante la rotación de las mismas. Su principal ventaja es que son fáciles de fabricar y pueden moverse a altas velocidades, lo que los hace adecuados para aplicaciones en las que se necesite cubrir grandes distancias en poco tiempo.

Uno de los tipos más comunes de rueda que se utilizan en los robots móviles son las ruedas Ackermann. Estas ruedas están diseñadas para permitir que el robot realice giros precisos sin necesidad de cambiar de dirección. Esto se logra mediante la utilización de un mecanismo de transmisión que permite que las ruedas delanteras giren en ángulos diferentes al girar el robot.

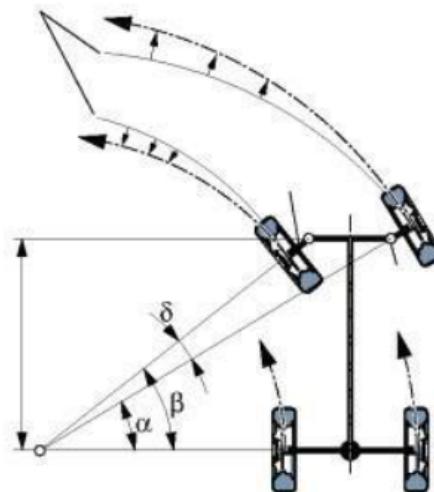


Figura 1: Esquema ruedas Ackermann

2. Objetivos

Para este trabajo se explicarán las diferentes configuraciones cinemáticas de robots móviles que existen, centrándose más en robots móviles que utilicen ruedas Ackermann. También se explicará el principio de Ackermann, y se pondrán ejemplos de robots actuales que utilicen estas ruedas, además de desarrollar una demo con ejemplos prácticos probados en Gazebo.

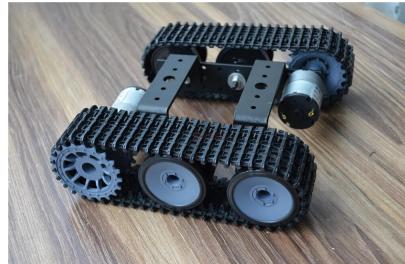
3. Configuraciones Cinemáticas de los Robots Móviles

La configuración cinética de un robot móvil se refiere a la forma en que está diseñado y se mueve. Algunas de las configuraciones más comunes de los robots móviles son:

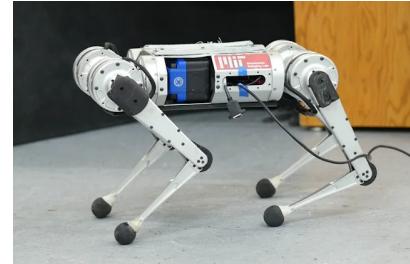
1. Robots con ruedas: las ruedas han sido el sistema de tracción y movimiento principal durante siglos, siendo utilizadas desde la antigüedad para mover cosas como carros y más recientemente en vehículos cotidianos. Este sistema de locomoción permite a los robots moverse por diferentes superficies, es fácil de controlar y utilizar, y puede alcanzar diferentes velocidades. Sin embargo, también presenta desventajas cuando se enfrentan a terrenos muy irregulares y complicados.
2. Robots con orugas: las orugas permiten que los robots se desplacen en terrenos suaves y les brindan una mayor capacidad de tracción debido a sus múltiples puntos de contacto en comparación con las ruedas. Sin embargo, también presentan algunas desventajas, como una menor velocidad y un menor control en comparación con las ruedas.
3. Robots con extremidades (patas): la robótica a menudo se inspira en los comportamientos y habilidades de los seres vivos. La locomoción mediante extremidades es común en la naturaleza y se observa en seres humanos, insectos y animales. En la robótica móvil, este tipo de configuración permite realizar maniobras que no son posibles con ruedas o orugas, pero también presenta una mayor complejidad en cuanto al diseño y control de los robots.



(a) Robot con ruedas



(b) Robot con orugas



(c) Robot con extremidades

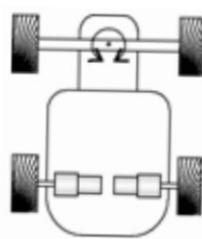
Figura 2: Tipos de configuraciones cinemáticas terrestres

3.1. Tipos de Robots Móviles con Ruedas

En los robots móviles con ruedas, se pueden utilizar diferentes configuraciones de ruedas que proporcionan al robot distintas características y propiedades en términos de eficiencia energética, dimensiones, capacidad de carga y maniobrabilidad. Algunas de estas configuraciones incluyen las ruedas Ackermann, las ruedas triciclo clásicas, la tracción diferencial, las ruedas skid steer, las ruedas síncronas y la tracción omnidiireccional. Cada una de estas configuraciones tiene sus propias ventajas y desventajas y es adecuada para diferentes aplicaciones. A continuación, se explicará por encima cada tipo de rueda:

3.1.1. Ruedas Ackermann

Están diseñadas para que las ruedas delanteras giren en diferentes ángulos durante un giro, de modo que el vehículo pueda mantener una trayectoria precisa y evitar patinar o derrapar. Esto es especialmente importante en vehículos como los coches de carreras y los autobuses, que necesitan un gran grado de control y estabilidad al girar en curvas cerradas.



(a) Ackerman



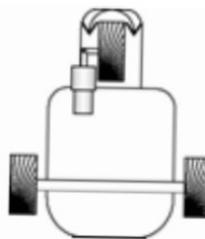
(a) Ackerman

(b) Ejemplo: Robomart

3.1.2. Ruedas Triciclo clásico

En este caso, el robot tiene tres ruedas dispuestas de manera similar a un triciclo infantil. Las dos ruedas traseras no están conectadas a ningún motor y pueden girar libremente. La rueda delantera se utiliza tanto para la tracción como para el direccionamiento. El cálculo de la odometría es más sencillo, ya que la posición del robot se determina por la cantidad de pulsos que avanza el encoder de la rueda motorizada y la dirección es simplemente la que lleva dicha rueda. Aunque este tipo de configuración ofrece una mayor maniobrabilidad, puede presentar problemas de estabilidad en superficies difíciles o irregulares.

Un problema común en estas ruedas es que el centro de gravedad del robot puede alejarse de la rueda de tracción en superficies inclinadas mientras el robot se desplaza hacia arriba. Esto puede resultar en una pérdida de tracción de la rueda hacia el suelo, lo que significa que la rueda motora sigue girando pero el robot no avanza. Esto puede causar un error acumulado significativo al hacer el seguimiento del desplazamiento del robot, ya que el robot puede informar que ha alcanzado su máximo desplazamiento cuando en realidad aún se encuentra mucho más atrás.



(b) Triciclo clásico

(a) Triciclo clásico

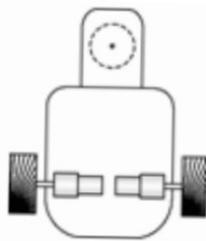


(b) Ejemplo:

3.1.3. Ruedas Direccionamiento diferencial

La configuración diferencial es la más simple de todas. Consiste en dos ruedas situadas diametralmente opuestas en un eje perpendicular a la dirección del robot, cada una de las cuales está accionada por un motor. Al cambiar la velocidad de cada motor, se puede hacer girar al robot a la derecha o a la izquierda. Sin embargo, con dos ruedas es imposible mantener el robot nivelado durante los giros, lo que puede causar movimientos bruscos o “cabeceos” al cambiar la dirección. Para solucionar este problema, se pueden añadir ruedas “locas” que no están accionadas por motores y giran libremente según la velocidad del robot.

Sin embargo, el uso de más de tres puntos de apoyo en el robot, incluyendo las dos ruedas de tracción, puede llevar a problemas de cálculo de odometría en terrenos irregulares y, en algunos casos, incluso a la pérdida total de tracción.



(c) Tracción diferencial

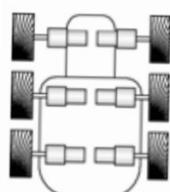


(a) Direcciónamiento diferencial

(b) Ejemplo: Roomba

3.1.4. Ruedas Skid Steer

En esta configuración, el robot está equipado con dos ruedas en cada lado que funcionan al mismo tiempo. Su movimiento se produce al cambiar las velocidades de las ruedas izquierda y derecha. Este tipo de robots se utiliza eficientemente en aplicaciones para exteriores, como la antropología, la inspección y la creación de mapas de lugares de difícil acceso, utilizando para ello un sistema de radar.



(d) Skid steer

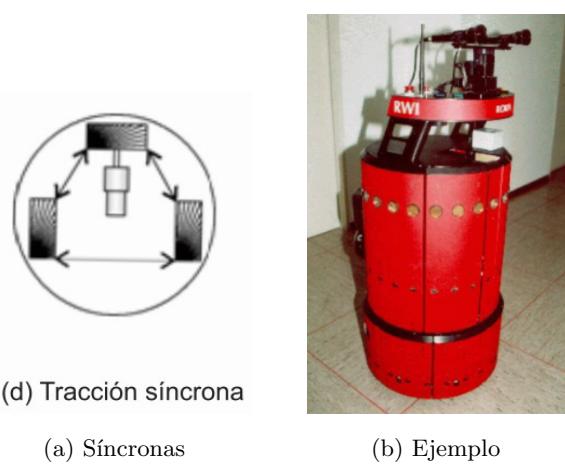


(a) Skid Steer

(b) Ejemplo: DeliveryBot

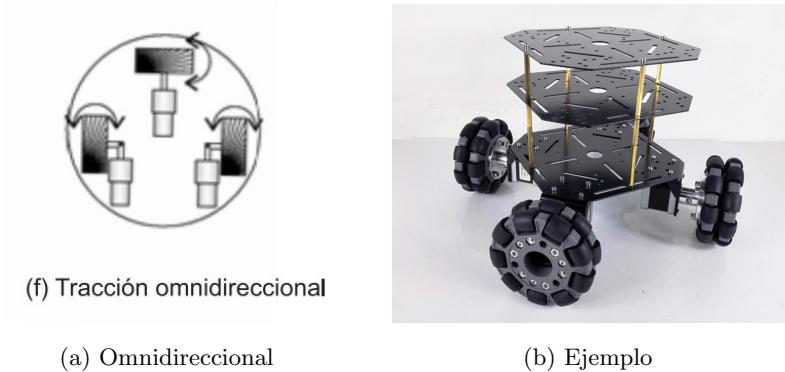
3.1.5. Ruedas Síncronas

Esta configuración consiste en una disposición innovadora que utiliza tres o más ruedas con tracción y acopladas mecánicamente de manera que todas giran a la misma velocidad y en la misma dirección. La transmisión se lleva a cabo mediante correas concéntricas. Esta configuración requiere una gran sincronización y resulta en una mejora en la odometría debido a que reduce el deslizamiento de las ruedas sobre el suelo, ya que todas las ruedas generan fuerzas con vectores de igual módulo y paralelos en todo momento.



3.1.6. Ruedas Tracción omnidireccional

Este tipo de sistema consta de tres ruedas que sirven tanto para guiar al robot como para proporcionarle tracción. Esta configuración tiene tres grados de libertad, lo que significa que puede realizar cualquier movimiento y posicionarse en cualquier orientación y lugar. No presenta restricciones cinemáticas, es decir, no hay limitaciones en cuanto al movimiento que el robot puede realizar.



4. ¿Qué es y cómo funciona el principio de Ackermann?

El principio de Ackermann establece que cuando un vehículo gira en una curva, los ejes de las ruedas deben converger en un punto, que es el centro de rotación instantáneo. Para que el vehículo pueda tomar la curva de manera eficiente y sin desgaste excesivo en las cubiertas, la rueda interior debe girar un ángulo mayor que la rueda exterior. Esto requiere que las ruedas delanteras diverjan ligeramente mientras el vehículo gira, lo que permite una geometría óptima en la dirección del vehículo. Como se muestra en la figura siguiente:

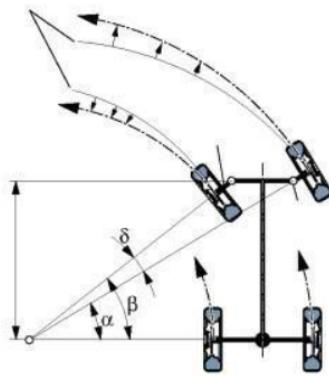


Figura 9: Concepto de Ackermann

Según Ackermann, el ángulo que forma la rueda exterior con la extensión del eje trasero α ha de ser menor que el formado por la interior β .

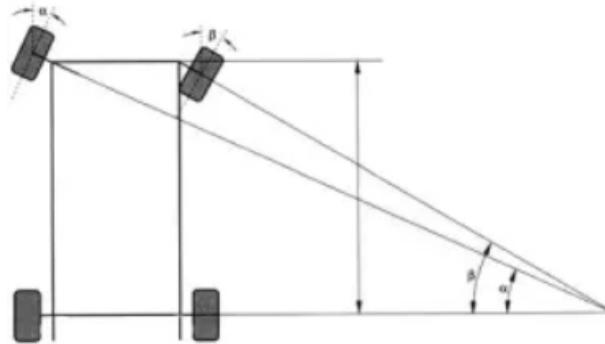


Figura 10: Ángulos de Ackermann

Si un vehículo es diseñado sin tener en cuenta el principio de Ackermann, y ambas ruedas delanteras giran con el mismo ángulo, no estarán girando en relación al mismo punto. Esto puede causar inestabilidad y un desgaste excesivo de los neumáticos. Es importante tener en cuenta este principio al diseñar un vehículo para garantizar una conducción segura y eficiente.

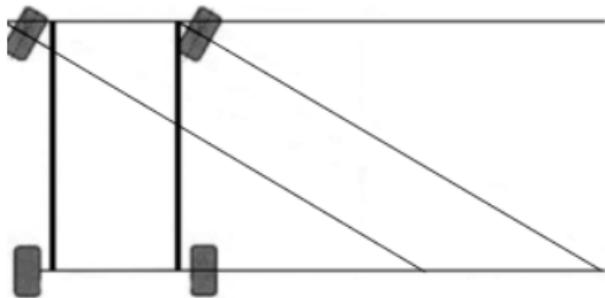


Figura 11: Vehículo sin principio de Ackermann

Cuando se aplica el principio de Ackermann, la rueda interior gira con un ángulo mayor para que todas las ruedas giren alrededor del mismo centro de rotación instantáneo. Esto se logra mediante la inclinación de las bieletas de dirección en relación al eje longitudinal, como se muestra en la siguiente imagen.

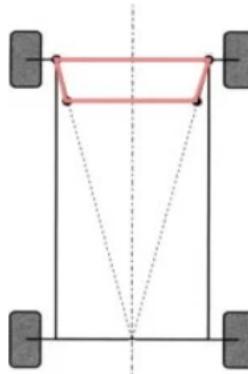


Figura 12: Inclinación de las bieletas

El cálculo de Ackermann se basa en la fórmula:

$$\frac{Via}{Batalla} = \cotg(\alpha) - \cotg(\beta) \quad (1)$$

La fórmula para calcular Ackermann se obtiene despejando α de la anterior, de modo que:

$$Ackermann = \arctan\left(\frac{batalla}{batalla/\tan(\beta) - via}\right) \quad (2)$$

$$Porcentaje = 100 * \frac{\alpha}{Ackermann} \quad (3)$$

Un 100 % de Ackermann significa que las prolongaciones de las bieletas se cortan en el centro del eje trasero. Si el porcentaje es mayor que 100 %, las prolongaciones se cortarán por delante del eje trasero, y si es menor, se cortarán detrás del eje trasero.

En teoría, el principio de Ackermann funciona de manera óptima, pero en la práctica, los neumáticos se deforman debido a las fuerzas que actúan sobre ellos. Esta deformación se conoce como ángulo de deriva, y representa la diferencia entre el ángulo de giro teórico y el ángulo real que adquiere la superficie de contacto del neumático debido a dichas fuerzas.

Los vehículos modernos no utilizan un sistema de dirección Ackermann puro debido a la transferencia lateral de masas durante la conducción. Esto significa que las ruedas que soportan menos carga necesitan menos ángulo de deriva para llegar a su límite de adherencia. Por esta razón, se utilizan sistemas de dirección más complejos que tienen en cuenta estas variables y permiten una conducción más segura y eficiente.

Para conseguir los efectos deseados en la geometría se debe introducir el concepto de convergencia.

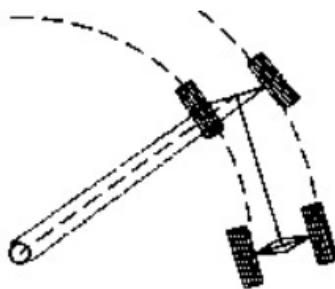


Figura 13: Concepto de convergencia

Cuando la convergencia es positiva, las dos ruedas directrices tienen cierta convergencia hacia el centro de las trayectorias. Esto significa que la rueda interior trata de describir una circunferencia ligeramente mayor y la rueda exterior una circunferencia ligeramente menor a la que está siguiendo. Esta geometría reduce el ángulo de deriva de la rueda interior y aumenta el ángulo de deriva de la rueda exterior.

Por otro lado, si la convergencia es negativa, la rueda interior trata de describir una circunferencia menor a la que está siguiendo y la rueda exterior una circunferencia mayor. Esto aumenta el ángulo de deriva de la rueda interior y reduce el ángulo de deriva de la rueda exterior.

5. Ejemplos de Robots que emplean las ruedas Ackermann

Como ya se ha mencionado anteriormente, las ruedas Ackermann son un tipo de dispositivo mecánico que se utiliza en algunos vehículos para mejorar la estabilidad y el control al girar. Esto es especialmente importante en vehículos como los coches de carreras y los autobuses, que necesitan un gran grado de control y estabilidad al girar en curvas cerradas. Además, se utiliza en muchos robots móviles.

Algunos de los robots móviles que utilizan estas ruedas de Ackermann son:

5.1. Curiosity

El robot móvil de exploración Mars Curiosity de la NASA. Este robot es conocido por sus grandes ruedas y su sistema de dirección Ackermann, que le permite moverse con precisión y estabilidad sobre el terreno accidentado del planeta rojo. Curiosity ha sido una plataforma de investigación clave para la NASA, ya que ha llevado a cabo numerosas pruebas y experimentos en busca de evidencia de agua y vida pasada en Marte.



Figura 14: Curiosity

5.2. Knightscope K5

Otro ejemplo de un robot móvil que utiliza ruedas Ackermann es el robot móvil de seguridad "Knightscope K5". Este robot es utilizado para vigilar y proteger edificios y otras instalaciones. Está equipado con una cámara de alta definición y una serie de sensores que le permiten detectar y monitorear actividades sospechosas. El robot también está diseñado para operar de manera autónoma y puede recorrer grandes áreas sin la necesidad de un operador humano. Las ruedas Ackermann le permiten al robot maniobrar en terrenos irregulares y seguir trayectorias precisas durante sus patrullas de seguridad.



Figura 15: Knightscope K5

5.3. Nomad

El Nomad fue desarrollado por MobileRobots, Inc. y utiliza cuatro ruedas Ackermann para su movilidad. Es un robot robusto y resistente, y está diseñado para operar en entornos difíciles, como terrenos irregulares, barro y arena. Además, es capaz de subir obstáculos y superar obstáculos en su camino. También cuenta con una variedad de sensores y equipos de navegación que le permiten mapear y explorar su entorno de manera efectiva.



Figura 16: Nomad en la Antártida

6. Aplicación práctica

Como ya se comentó anteriormente, algunos de los ejemplos más comunes en los que se utilizan las ruedas tipo Ackerman es en coches de carreras, es por eso que se ha desarrollado la siguiente aplicación práctica.

6.1. Race car

La idea para este proyecto era desarrollar un paquete en ROS en el cual se utilizase un robot móvil que siguiese el principio Ackermann para moverse. Para ello se realizó un trabajo de investigación para encontrar información acerca de este principio y como poder usarlo para desarrollar en ROS. Se encontró dentro de la documentación oficial una página que proporciona enlaces a todo el código ROS actual relacionado con los robots basados en la dirección de Ackermann. En él se encuentran diferentes paquetes que se pueden usar para la creación de mensajes ROS para estos vehículos o para implementar pluggins que mejoran la pila de navegación, entre otros muchos. [4] [5]

Además se buscó algún paquete ya creado en ROS sobre el cual basar nuestro proyecto y se encontró un github muy extenso con un proyecto similar. [7]

Para este proyecto se han utilizado algunas de las implementaciones ya creadas de ese paquete, pero se han añadido cambios para poder realizar más tareas o suprimir aquellas que no interesaban. El contenido final del paquete es el que se describe a continuación. El paquete está disponible en Github para poder descargarlo y probarlo [8].

- **racecar_control:** En esta carpeta se encuentran definidos los diferentes controladores que utilizará el robot. De esta manera se podrá publicar la información articular en Gazebo y simular su movimiento correctamente. Así como un launch para lanzarlos y un script que controla el movimiento del robot.
 - **config:** Dentro se encuentra un fichero .yaml donde se define en primer lugar el controlador JointStateController, que es un controlador genérico usado por el nodo controller_manager para publicar el estado de todas las articulaciones. Después se definen los controladores de posición y velocidad como effort_controllers que son otro tipo de controladores los cuales publican el par necesario para mover la articulación y necesitan ajustar correctamente el PID para su funcionamiento.
 - **launch:** Contiene un fichero .launch para cargar las configuraciones definidas en el fichero .yaml explicado en el punto anterior y lanza el nodo controller_manager. Además también lanza el nodo RobotStatePublisher que es una herramienta muy útil que permite referenciar todos los TF (otra herramienta que relaciona de manera sencilla los diferentes sistemas de coordenadas de los sensores, articulaciones y robots que aparecen en ROS) entre sí sin necesidad de ir publicando manualmente los TF de todas las articulaciones. También se lanzan dos nodos más que sirven para poder el mover el robot adecuadamente.
 - **scripts:** Esta es una carpeta que contiene el script servo_commands el cual es utilizado para publicar en los topics correspondientes la velocidad y el ángulo de giro de cada rueda por separado.
- **racecar_description:** En ésta se encuentran los archivos que definen como es el robot que se ha diseñado. Cuenta con tres subcarpetas:
 - **urdf:** Contienen los archivos URDF donde viene descrito cómo es físicamente el robot con el que se está trabajando. Se indican los eslabones (sus medidas, pesos, formas, fricciones...), las articulaciones (donde se definen el origen, si tienen límites, qué tipo de articulación es...) y las transmisiones.

- **meshes:** Aquí están guardadas figuras más complejas creadas con una forma específica mediante un software de diseño.
- **models:** En ésta se encuentran definidos los modelos de configuración y visualización de otros objetos que se utilicen durante la simulación en Gazebo. En este caso solo está creado el modelo de un cono.
- **racecar_gazebo:** Aquí se guardan los archivos que nos permiten lanzar la simulación en Gazebo y Rviz. Así como los diferentes mundos donde se puede mover el robot y los scripts que permiten desplazarlo, como por ejemplo mediante la teleoperación haciendo uso de las teclas u otro código que permite al robot moverse siguiendo el camino generado por el planificador de trayectorias de rviz. También se encuentran diferentes launch para realizar distintas tareas con el robot como el mapeado del mundo o la navegación autónoma. Y una carpeta adicional donde se han guardado los mapas que creados durante la etapa de mapeado.
 - **config:** En esta carpeta se encuentran varios archivos .yaml que contienen parámetros predefinidos relacionados con el movimiento del robot. Además de varios entornos de rviz ya configurados con las funcionalidades necesarias para que cuando sean llamados puedan ser utilizados desde el principio.
 - **launch:** Contiene los launches para lanzar un mundo en Gazebo, abrir otra terminal en rviz y mapearlo, para después poder navegar en él utilizando las teclas (w,a,s,d) o para lanzar un script que mueva al robot siguiendo una trayectoria.
 - **map:** En ésta están almacenados los diferentes mapas de los mundos que se han guardado como un fichero .pgm después de haber hecho el mapeado.
 - **scripts:** Aquí están los 3 scripts que se pueden utilizar para mover el robot. Los dos primeros, gazebo_odometry.py y keyboard_teleop.py, se llaman automáticamente cuando lanzas el launch que abre gazebo. Y el tercero se puede utilizar para que el robot siga la trayectoria que defina rviz cuando se haga uso de la herramienta 2D Nav Goal.
 - **worlds:** Esta última carpeta contiene los mundos que se han utilizado para las simulaciones. Dos de ellos creados desde cero para este proyecto y el otro era el que se utilizaba en el repositorio que se ha utilizado como punto de partida.
- **system:** Es una carpeta que contiene archivos que se llaman internamente a la hora de realizar el control de nuestro robot. Sin ellos, a la hora de lanzar el paquete existirían problemas de dependencias.

6.1.1. Creación del mundo

A la hora de la simulación se ha elegido Gazebo por su capacidad para simular todo tipo de robots, sensores y objetos en un mundo tridimensional. Además, es destacable su capacidad para generar una retroalimentación realista de los sensores cuando se interacciona con objetos del mapa digital, por lo que debido a que este proyecto no se va a probar en el laboratorio, se ha considerado ejecutar la simulación más realista posible.

Para simular en Gazebo, lo primero que se necesita es un mundo. En Github existen mundos ya creados por otros usuarios que se pueden descargar y usar, pero para hacer el proyecto más personalizado se ha decidido crear desde cero dos opciones diferentes:

1. La primera es una opción simple compuesta por una pista cuadrada y un cono de tráfico. Esta se ha usado para ejecutar las pruebas de movimiento y comprobar el funcionamiento general.

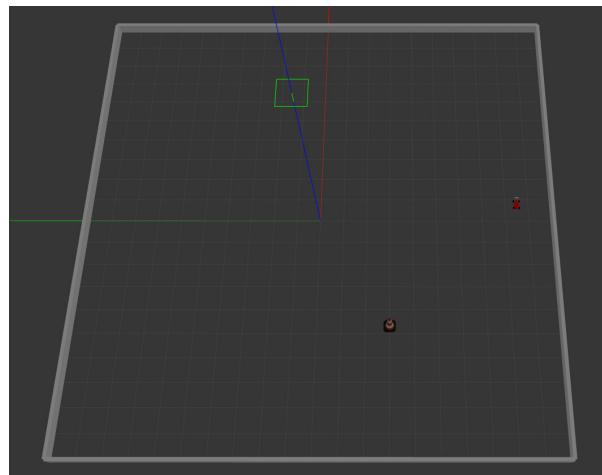
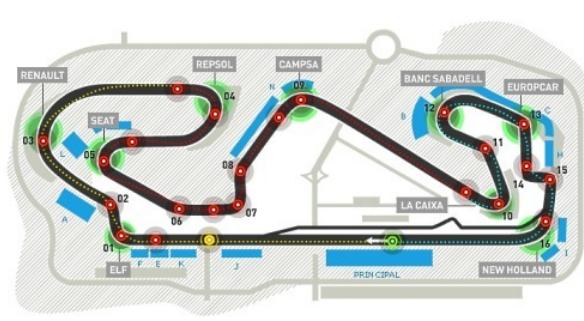
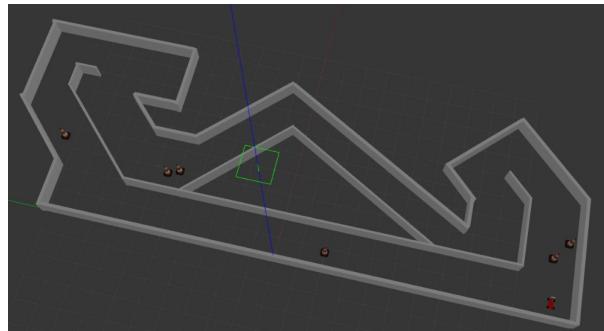


Figura 17: Circuito simple

2. La segunda es una pista similar a las de carreras. En concreto se ha diseñado de forma similar al circuito de Montmeló (Barcelona). Esta se ha usado para probar las trayectorias más concretas, refinar los movimientos y hacer pruebas más precisas.



(a) Circuito original de Montmeló



(b) Circuito de Gazebo

Figura 18: Circuito complejo

Los mundos son archivos con extensión .world escritos en lenguaje xml en los que se especifican las características del suelo, los elementos del mundo, la geometría de los objetos, su posición y orientación, la iluminación y la física del mundo. Para construir ambos mundos se ha seguido el mismo procedimiento, ya que esencialmente están compuestos por los mismos componentes.

Respecto a las paredes se deben especificar tanto el área de colisión como los rasgos visuales. Dentro del primer aspecto, se definen características como el tipo de superficie y el tipo de fricción. En el segundo aspecto se elige como es el material que se desea ver, el color o la textura. Además, en ambos hay que definir el tamaño y posición de la pieza o área que se está tratando. En el caso de ambos circuitos, todas las paredes están creadas con los siguientes parámetros:

- Número de contactos: 10. Este valor es el número máximo de puntos de contactos que se van a generar en cada colisión. Como en este caso la idea es que no haya choques, el número es pequeño, pero en caso de querer realizar algún otro proyecto, el aumento de los puntos de contacto aumentaría la precisión del choque. Es

útil definirlo de antemano para reservar una cantidad de memoria fija y no tener que hacer una resignación durante la ejecución.

- Fricción y contacto: se utiliza un parámetro de la biblioteca ODE (Open Dynamics Engine) para simular el comportamiento físico de objetos en movimiento.
- Material: Gazebo gris. Otras opciones son otros colores, colores metalizados, colores transparentes o diferentes estampados como césped, madera, ladrillos.... [6]
- Ambiente: <1 1 1 1>. Se corresponden a los valores RGBA (rojo, verde, azul y alfa). En este caso el color de la luz ambiental sería blanco, ya que todos los valores son 1. La luz ambiental es una luz que se aplica a todos los objetos en una escena, y suele utilizarse para simular la luz indirecta y difusa que se refleja en los objetos.
- Tamaño: Para cada una de las paredes se ha definido el mismo tamaño para el área visual que para el área de colisión.

Otra parte del relieve de estos mapas son los conos de tráfico. Este tipo de piezas no están incluidos en Gazebo, por lo que para utilizarlas se ha tenido que cargar un archivo 3D con el diseño. La cantidad de parámetros configurables es similar a lo que se ha explicado anteriormente, añadiendo además una etiqueta de escala.

6.1.2. Creación del robot con ruedas Ackermann

El robot, con apariencia de coche en este caso, está creado como un archivo URDF para poder configurar cada una de las piezas y las relaciones entre ellas. Para una simulación más precisa y detallada del comportamiento mecánico, se ha descargado de un repositorio de Github [7] una implementación que cumplía con los requisitos del robot que se quería desarrollar.

Respecto a la geometría, está formado por un chasis y cuatro ruedas cargadas como distintos modelos 3D STL. Además, tiene un sensor laser Hokuyo y una cámara ZED. La apariencia del coche es la siguiente:

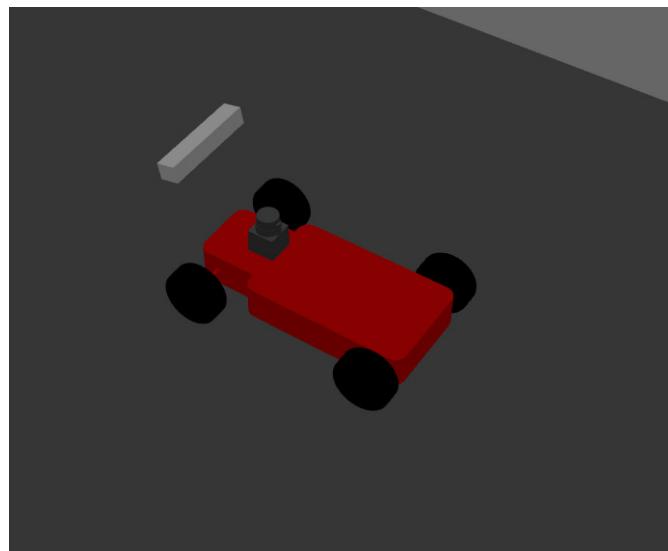


Figura 19: Coche con ruedas Ackermann

En el archivo macros.xacro se encuentra la información referente al peso que se le asigna a cada uno de los componentes, la posición en la que se encuentran, sus valores de inercia, el tipo de transmisión entre las ruedas y la

ubicación de cada archivo STL.

En el archivo racecar.gazebo están definidos los plugins de la cámara y el láser, con sus correspondientes parámetros configurables como la posición, la resolución, el color, sus controladores...

A la hora de mover el coche, se han desarrollado dos opciones diferentes. En primer lugar, se puede mover teleoperándolo con las teclas WASD. En segundo lugar, se puede hacer uso del planificador de trayectorias de RVIZ y enviarlo a un destino deseado. Para esta última opción es importante que como este robot es externo a los que conoce ROS, no puede utilizar el paquete de navegación que solemos usar con los turtlebots. Para ello, ha sido necesario desarrollar un script que lea los puntos del camino desarrollado por la pila de navegación, y publique en los topics de cada rueda su información correspondiente relativa al ángulo de giro y la velocidad. El script desarrollado es el que se explica en el siguiente apartado.

6.1.3. Desarrollo del script path_pursuit.py

Para poder mover el robot sin hacer uso del keyboard_teleop.py y conseguir que el robot se mueva siguiendo unas trayectorias determinadas se ha elaborado un script con ayuda de las diferentes referencias encontradas. [9] [10]

La idea era utilizar la pila de navegación 2D para iniciar la localización y así poder enviarle los objetivos al robot. En rviz está la herramienta 2D Nav Goal que permite al usuario establecer una pose deseada a la navegación y que el robot lo logre. Esto se consigue gracias a que la pila de navegación toma información de la odometría, flujos de los sensores y esa pose objetivo para generar comandos de velocidad seguros que se envían a una base móvil.

Sin embargo, el uso de la pila de navegación en un robot arbitrario es más complicado de utilizar. Como requisito previo para su uso el robot debe ejecutar ROS, tener un árbol de transformación tf y publicar los datos del sensor utilizando los tipos de mensajes ROS correctos. Además, la pila de navegación debe configurarse para que la forma y la dinámica del robot funcionen a alto nivel.

Teniendo en cuenta todo esto y diseñados los archivos necesarios ya se pudo comenzar a escribir el código. En primer lugar hay que suscribirse a `/pf/pose/odom` para obtener la información de la odometría y así acceder a la pose actual del robot. Y también hay que suscribirse a `/move_base/TebLocalPlannerROS/global_plan` que nos proporciona la secuencia de puntos de la trayectoria a seguir creado por el planificador. Además para publicar las velocidades y giros adecuados debemos de hacerlo en el topic `/vesc/low_level/ackermann_cmd_mux/input/navigation`.

La suscripción a los topics nos permite recibir los mensajes y operar con la información que contienen mediante dos callbacks: `callback_read_path` y `callback_read_current_position`. El primero nos permite leer en tiempo real los puntos de la trayectoria generada. Este camino será guardado en una variable global para ser usado por `callback_read_current_position`. Éste otro nos permite leer la posición actual del coche y, conociendo los diferentes puntos del path, se buscaría el más cercano al vehículo para definir un objetivo local (diferente al punto objetivo final) y calcular la curvatura que debería seguir el coche y los ángulos de giro de las ruedas. Para finalizar, es necesario escribir la velocidad y el ángulo como un mensaje del tipo `ackermann_msgs/AckermannDriveStamped` y publicarlo en el topic correspondiente.

6.1.4. Ejecución del paquete

Este paquete ha sido probado con éxito en Ubuntu 20.04 usando ROS NOETIC.

Para comenzar, las dependencias necesarias que se necesitan son las siguientes:

```
sudo apt install ros-$ROS_DISTRO-gazebo-ros-control
sudo apt install ros-$ROS_DISTRO-effort-controllers
sudo apt install ros-$ROS_DISTRO-joint-state-controller
sudo apt install ros-$ROS_DISTRO-driver-base
sudo apt install ros-$ROS_DISTRO-ackermann-msgs
sudo apt install ros-$ROS_DISTRO-rtabmap-ros
sudo apt install ros-$ROS_DISTRO-teb-local-planner

sudo apt install tcl-dev tk-dev python3-tk
```

El primer paso para probar el coche de carreras es creando un nuevo workspace. Luego, en la carpeta src, se debe descomprimir el paquete de race_car.

```
cd ~/racecar_ws/src
git clone https://github.com/Luperbal/Race-Car.git
cd ..
catkin_make
```

Una vez hecho esto, el paquete está listo para usarse.

Si el mapa no está guardado, es necesario **mapear el entorno**. Para ello hay que abrir el mundo en Gazebo:

```
cd ~/racecar_ws/src
catkin_make
source devel/setup.bash
roslaunch racecar_gazebo racecar_runway.launch
```

Para moverse hay que tener abierta la ventana pequeña y utilizar las teclas WASD.

En una nueva terminal, se abre RVIZ, que es el que va **generar el mapa** del mundo.

```
cd ~/racecar_ws/src
catkin_make
source devel/setup.bash
roslaunch racecar_gazebo slam_gmapping.launch
```

Una vez mapeado por completo **se guarda el mapa** con el siguiente comando. Para ello también hay que abrir una nueva terminal:

```
cd ~/racecar_ws/src
catkin_make
source devel/setup.bash
rosrun map_server map_saver -f ~/racecar_ws/src/racecar/racecar_gazebo/map/mapa
```

Para comenzar la **navegación**, se vuelve a abrir Gazebo:

```
cd ~/racecar_ws/src
catkin_make
source devel/setup.bash
roslaunch racecar_gazebo racecar_runway_navigation.launch
```

En segundo lugar, se abre RVIZ en una nueva terminal. Haciendo click en “**2D Nav Goal**” y en la posición deseada del mapa, se genera el **camino que seguirá el robot**.

```
cd ~/racecar_ws/src
catkin_make
source devel/setup.bash
roslaunch racecar_gazebo racecar_rviz.launch
```

Finalmente, para que vaya a esa posición se llama al script de **path_pursuit** desde una nueva terminal:

```
cd ~/racecar_ws/src
catkin_make
source devel/setup.bash
rosrun racecar_gazebo path_pursuit.py
```

6.1.5. Vídeo

A continuación se adjunta el enlace a un vídeo https://youtu.be/IRxWt20X_rQ, el cual explica de forma un poco más dinámica y divertida los conceptos más importantes del principio de Ackermann y la aplicación práctica realizada del paquete de race_car.

6.2. Simulación de Curiosity

Una vez creado todo este proyecto, se ha querido ver cual es el nivel de complejidad que tienen otros robots reales que hacen uso de ruedas Ackerman. Por ello se ha simulado también el robot Curiosity en Gazebo, utilizando un modelo 3D preciso del robot y simulando su comportamiento físico y los sensores que tiene a bordo. En su simulación se ha utilizado un entorno marciano “The Construct Sim”. El proceso de funcionamiento es el siguiente:

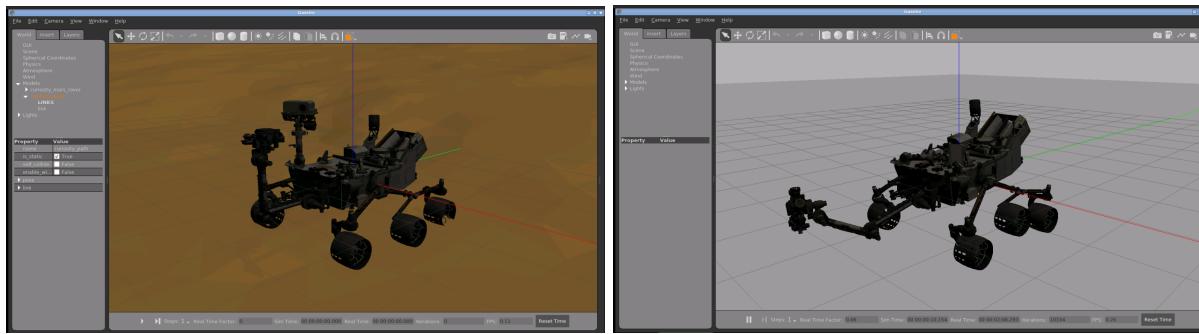
Primeramente, se crea el espacio de trabajo:

```
cd ~/catkin_ws/src
git clone https://github.com/ArghyaChatterjee/curiosity_mars_rover_description.git
cd ~/catkin_ws
catkin_make
```

Seguidamente, se lanza la simulación:

```
cd catkin_ws
source devel/setup.bash
roslaunch curiosity_mars_rover_description main_real_mars.launch
```

Si todo va bien, se deberían abrir dos ventanas, una con la simulación del mundo de Marte en Gazebo y otra con el modelo del Curiosity en Rviz.



(a) Simulación en Gazebo

(b) Simulación en Rviz

Figura 20: Simulación Curiosity

Tras mucho tiempo de simulación, se consiguió cargar la cámara del robot en Rviz y así poder ver el mundo marciano de Gazebo.

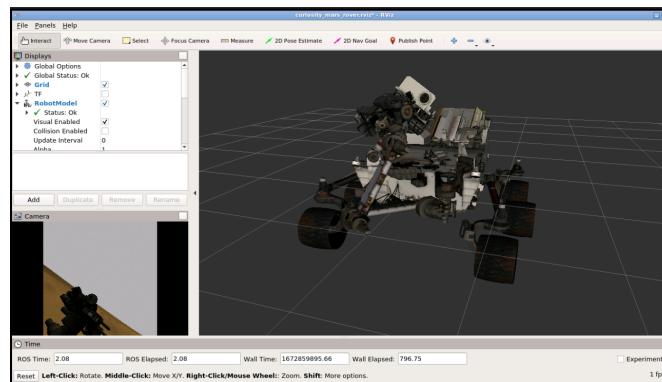


Figura 21: Cámara de Curiosity

Como se observa en la imagen superior, la cámara del robot se encuentra ladeada, y abajo a la izquierda está la vista del mundo en Gazebo. También se observa abajo a la derecha que la simulación iba a 1fps es por eso que apenas se consiguió mover el robot. No obstante, se adjunta a continuación un pequeño vídeo demostrativo de como debería verse la simulación de este robot: https://youtu.be/0MgeNCWg_CY

Lo interesante aquí, es ver como giran las ruedas, fijarse en como las dos ruedas delanteras en un giro tienen un ángulo diferente, siendo mayor el ángulo de la rueda interior del giro, como se ha explicado en el apartado del “Principio de Ackermann”.

7. Conclusión

En conclusión, este trabajo ha tenido resultados muy satisfactorios tanto en el ámbito teórico como práctico, aprendiendo multitud de conceptos importantes en el campo de la robótica móvil. Se ha demostrado la eficacia de las ruedas de Ackermann en un entorno práctico utilizando el sistema operativo ROS y creando prácticamente cada elemento de los que se usan desde cero. Los resultados obtenidos han sido satisfactorios y han demostrado la capacidad de las ruedas de Ackermann para mejorar la estabilidad y la precisión en la navegación del robot, sobre todo a la hora de maniobrar en espacios reducidos con una mayor precisión y estabilidad que otros robots con otros tipos de ruedas.

8. Bibliografía

Referencias

- [1] Configuraciones cinemáticas de robots móviles: <https://openwebinars.net/blog/robotica-movil-que-es-y-sus-aplicaciones/#:~:text=En%20los%20robots%20m%C3%B3viles%20con,eficiencia%20energ%C3%A9tica%2C%20dimensiones%2C%20cargas%20y>
- [2] Tipos de ruedas en robots móviles: <https://library.co/article/veh%C3%ADculos-ruedas-configuraciones-cinem%C3%A1ticas-robots-m%C3%B3viles.qv15m541>
- [3] Principio de Ackermann: <https://www.ingenieriamecanicaautomotriz.com/que-es-y-como-funciona-el-principio-de-Ackermann/>
- [4] Documentación paquete navegación: <http://wiki.ros.org/Ackermann%20Group>
- [5] Documentación paquete navegación: http://wiki.ros.org/ackermann_steering_controller
- [6] Listado de materiales disponibles en Gazebo: http://wiki.ros.org/simulator_gazebo/Tutorials/ListOfMaterials
- [7] Documentación paquete navegación: https://github.com/linklab-uva/f1tenths_gtc_tutorial
- [8] Github paquete race_car: <https://github.com/Luperbal/Race-Car.git>
- [9] Documentación paquete navegación: <http://wiki.ros.org/navigation>
- [10] Setup and Configuration of the Navigation Stack on a Robot <http://wiki.ros.org/navigation/Tutorials/RobotSetup>
- [11] Repositorio Github de Curiosity: https://github.com/ArghyaChatterjee/curiosity_mars_rover_description
- [12] Vídeo simulación de Curiosity en Gazebo: https://youtu.be/0MgeNCWg_CY